

Developer
Experience is more
than just Productivity
metrics



DevEx disasters...

Bad DevEx

Common examples

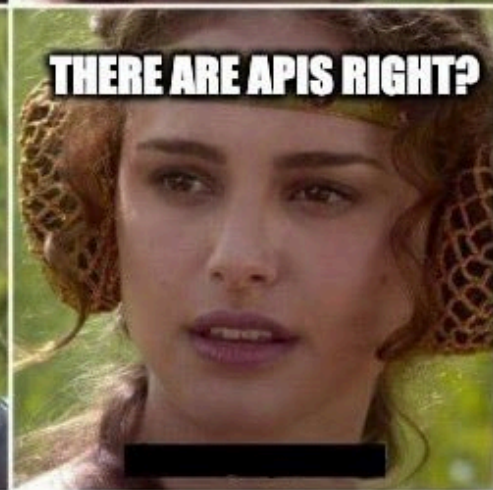
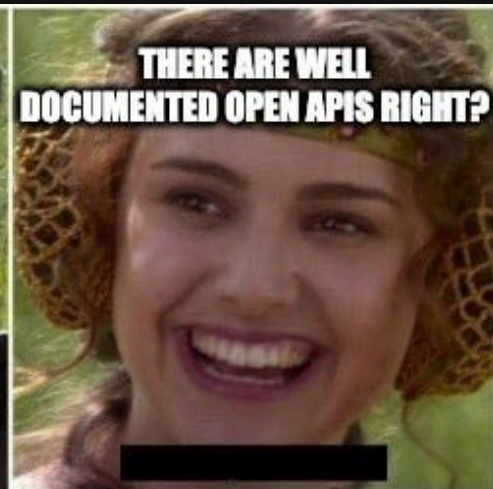
- Poorly documented features (or bugs)



Bad DevEx

Common examples

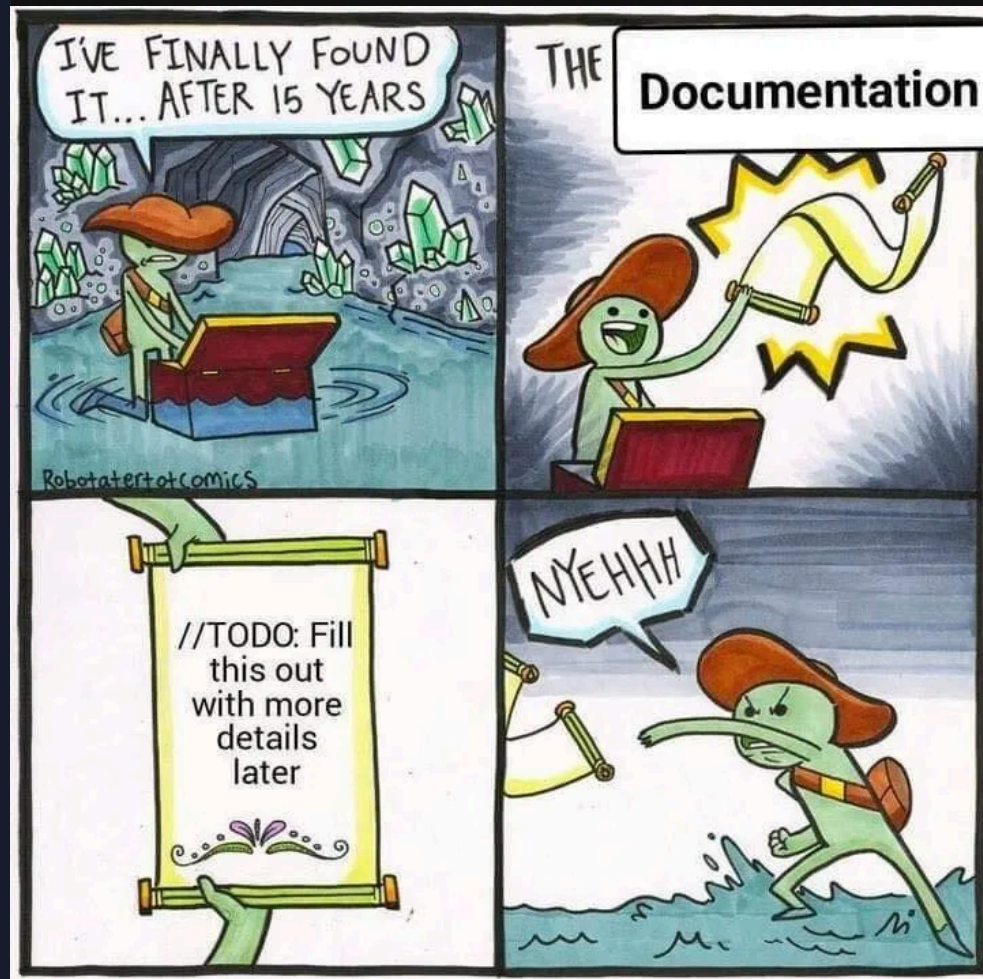
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)]



Bad DevEx

Common examples

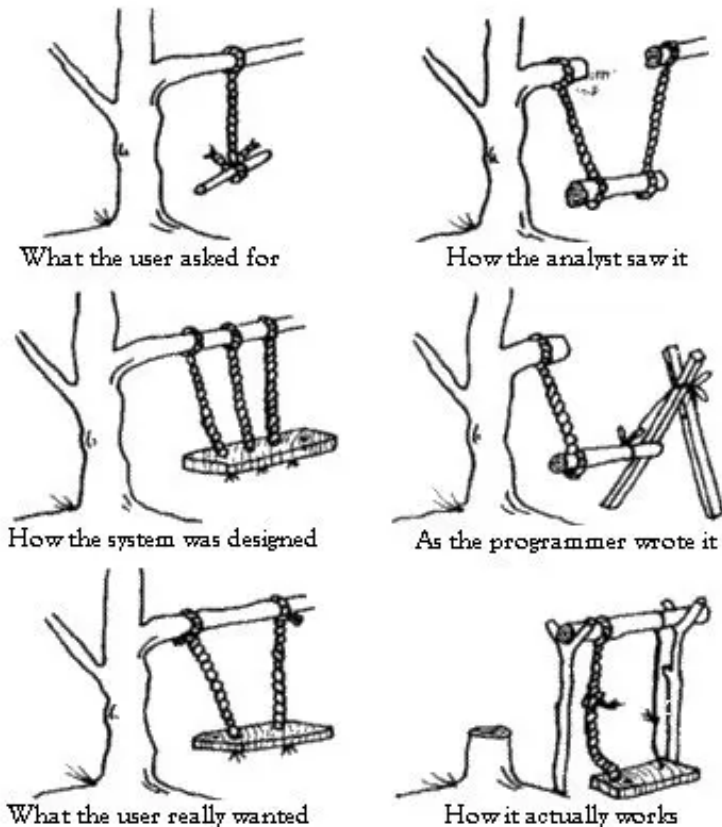
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- PDF documentation... or access-gated



Bad DevEx

Common examples

- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- PDF documentation... or access-gated
- Missing examples... of *anything*




Bad DevEx

Common examples

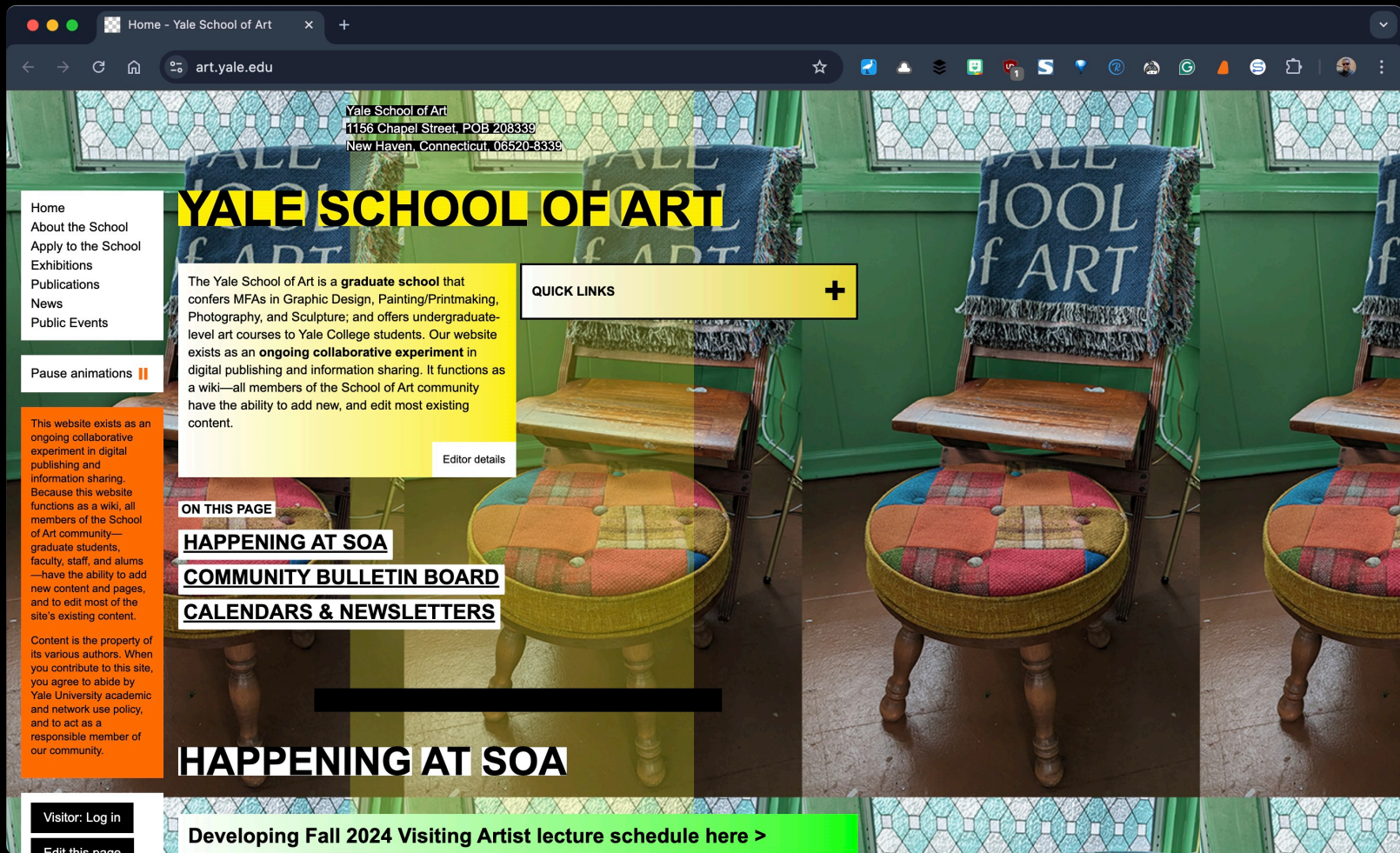
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- PDF documentation... or access-gated
- Missing examples... of *anything*
- "CI as Magic 8-Ball"



Ramiro Berrelleza 
@rberrelleza

Long time ago, in a galaxy far away, I worked at a team where our CI environment was so different from local or production, that the only realistic option way to validate a change was in prod. So we would commit the change, rerun CI jobs until they were green, deploy to prod, and then monitor the logs for about 1 hour. If no major errors were logged after that you were good to go 🙄

12:39 AM · Aug 3, 2024



Yale School of Art
1156 Chapel Street, POB 208339
New Haven, Connecticut, 06520-8339

YALE SCHOOL OF ART

Home
About the School
Apply to the School
Exhibitions
Publications
News
Public Events

Pause animations 

This website exists as an ongoing collaborative experiment in digital publishing and information sharing. Because this website functions as a wiki, all members of the School of Art community—graduate students, faculty, staff, and alumni—have the ability to add new content and pages, and to edit most of the site's existing content.

Content is the property of its various authors. When you contribute to this site, you agree to abide by Yale University academic and network use policy, and to act as a responsible member of our community.

The Yale School of Art is a **graduate school** that confers MFAs in Graphic Design, Painting/Printmaking, Photography, and Sculpture; and offers undergraduate-level art courses to Yale College students. Our website exists as an **ongoing collaborative experiment** in digital publishing and information sharing. It functions as a wiki—all members of the School of Art community have the ability to add new, and edit most existing content.

[Editor details](#)

QUICK LINKS



ON THIS PAGE

[HAPPENING AT SOA](#)

[COMMUNITY BULLETIN BOARD](#)

[CALENDARS & NEWSLETTERS](#)

HAPPENING AT SOA

Visitor: [Log in](#)

[Edit this page](#)

[Developing Fall 2024 Visiting Artist lecture schedule here >](#)

Yale School of Art
1156 Chapel Street, POB 208338
New Haven, Connecticut 06520-8338

Home
About the School
Apply to the School
Exhibitions
Publications
News
Public Events

Pause animations ||

This website exists as an ongoing collaborative experiment in digital publishing and information sharing. Because this website functions as a wiki, all members of the School of Art community—graduate students, faculty, staff, and alumni—have the ability to add new content and pages, and to edit most of the site's existing content.

Content is the property of its various authors. When you contribute to this site, you agree to abide by Yale University academic and network use policy, and to act as a responsible member of our community.

Visitor: Log in

Edit this page

Page history

YALE SCHOOL OF ART

The School of Art is one of **Yale's Graduate & Professional Schools** conferring MFA degrees in Graphic Design, Painting/Printmaking, Photography, and Sculpture; and offers undergraduate-level art courses to Yale College students. Our website exists as an **ongoing collaborative experiment** in digital publishing and information sharing. It functions as a wiki—all members of the School of Art community have the ability to add new, and edit most existing content.

Editor details

QUICK LINKS



ON THIS PAGE

HAPPENING AT SOA

COMMUNITY BULLETIN BOARD

CALENDARS & NEWSLETTERS

HAPPENING AT SOA

See the developing Spring 2025 Visiting Artist lecture schedule [here >](#)

[Fall 2024 archive >](#)

```
git push heroku main
```



Deploy to Heroku

Jeremy Meiss

Director, DevEx & DevRel

OneStream Software

DevOpsDays KC Organizer



Developer Experience vs Developer Productivity

Developer Experience

_"...the **journey** of developers as they learn and deploy technology, which if successful, focuses on eliminating obstacles that hinder a developer or practitioner from achieving success in their endeavors."

-**Jessica West**, Director of Education & Customer Experience (Chronosphere)

Developer Productivity

"Developer productivity refers to the effectiveness and efficiency with which software developers produce high-quality code and complete projects.""

■ LinearB



Developer Experience != Developer Productivity

♪ More than a metric.... ♪

Developer Experience != Developer Productivity

Experience is the **Cause**



The Roots: DevEx

The daily journey: tools, processes, cognitive load, and flow state.

Productivity is the **Effect**



The Fruit: Productivity

The outcome: high-quality, impactful software that drives business value.



Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](#)

Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452 \[cs.SE\]](#)

(or [arXiv:1312.1452v1 \[cs.SE\]](#) for this version)

<https://doi.org/10.48550/arXiv.1312.1452> 

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition," 2012 International Conference on Software and System Process (ICSSP), Zurich, Switzerland, 2012.



Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](http://this.http.URL)

Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452](https://arxiv.org/abs/1312.1452) [cs.SE]

(or [arXiv:1312.1452v1](https://arxiv.org/abs/1312.1452v1) [cs.SE] for this version)

<https://doi.org/10.48550/arXiv.1312.1452> 

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

"New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments."

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition. 2012."




Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](http://this)

Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452](https://arxiv.org/abs/1312.1452) [cs.SE]
(or [arXiv:1312.1452v1](https://arxiv.org/abs/1312.1452v1) [cs.SE] for this version)
<https://doi.org/10.48550/arXiv.1312.1452> 

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

"...developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance."

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition. 2012."

From Lines of Code to Value Streams

!!!

Leaders agree that traditional metrics like LOC are ineffective.

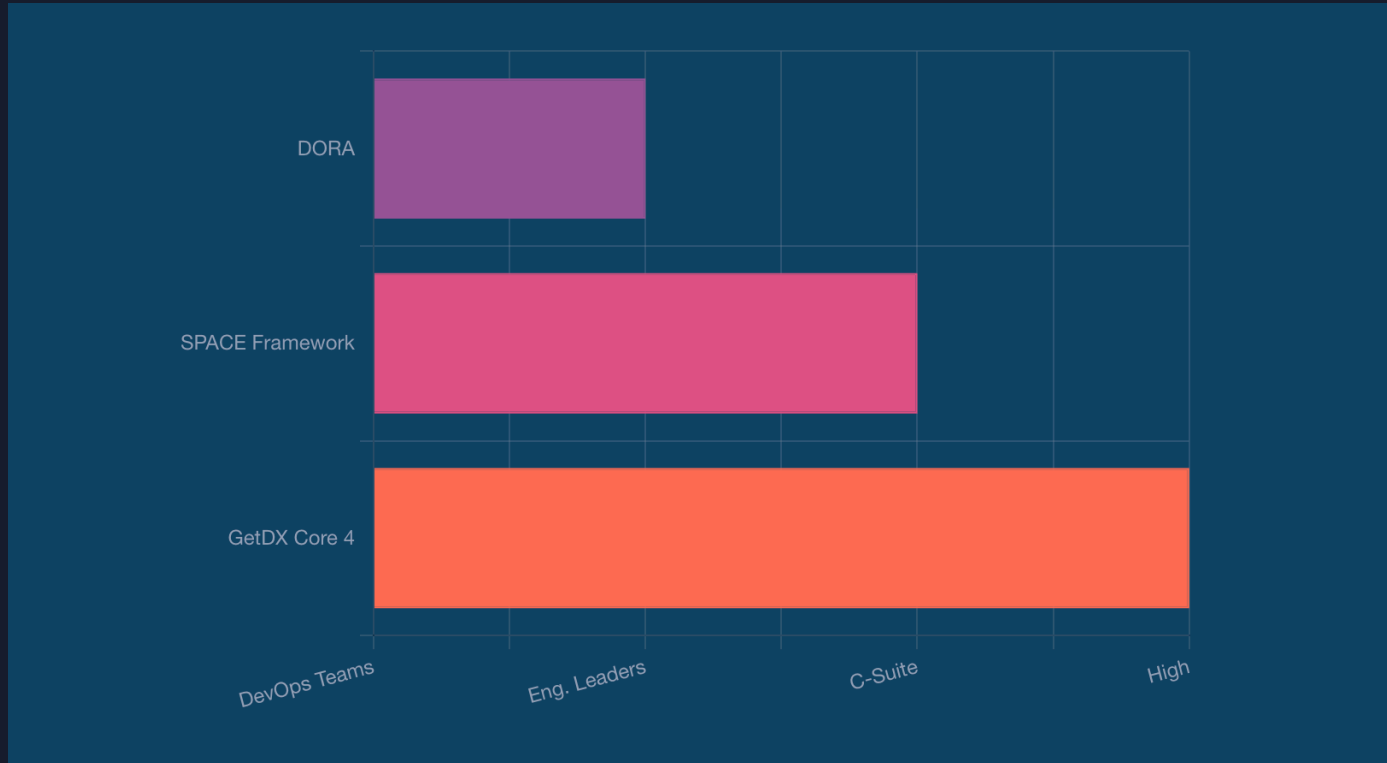
1st

Principle of modern measurement:
Focus on systems, not just individuals.

3

Prominent frameworks now guide the industry: DORA, SPACE, and GetDX Core 4.

Evolution of "Developer Experience" Frameworks



The Evolutionary Path of Measurement

2014-2018: DORA Metrics Emerge

DORA established the gold standard for measuring DevOps pipeline health through rigorous research, proving speed & reliability are not trade-offs...

2021: The SPACE Framework Broadens the Scope

SPACE framework introduced a holistic, human-centric model, arguing that productivity is multi-dimensional, emphasized the importance of developer satisfaction, well-being, and collaboration as critical components of performance...

2024: GetDX Core 4 Unifies for Business Impact

GetDX Core 4 was created to be a practical, prescriptive framework attempting to unify DORA and SPACE, creating a link between engineering efforts and tangible business outcomes like ROI and revenue.

Deep Dive: DORA's Four Keys

Deployment Frequency

How often code is deployed to production. High frequency indicates agile processes and small batch sizes.



Lead Time for Changes

The time it takes for a commit to get into production. Short lead times signify efficient CI/CD pipelines.



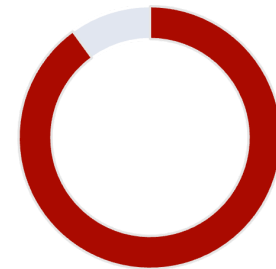
Change Failure Rate

The percentage of deployments causing a failure in production. A low rate indicates high quality and robust testing.



Time to Restore Service

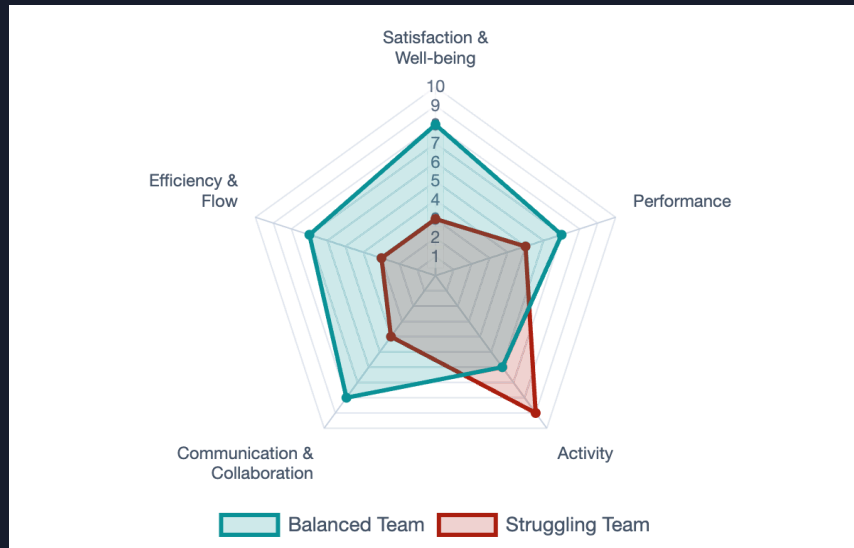
How long it takes to recover from a failure in production. A low time indicates high system resilience.



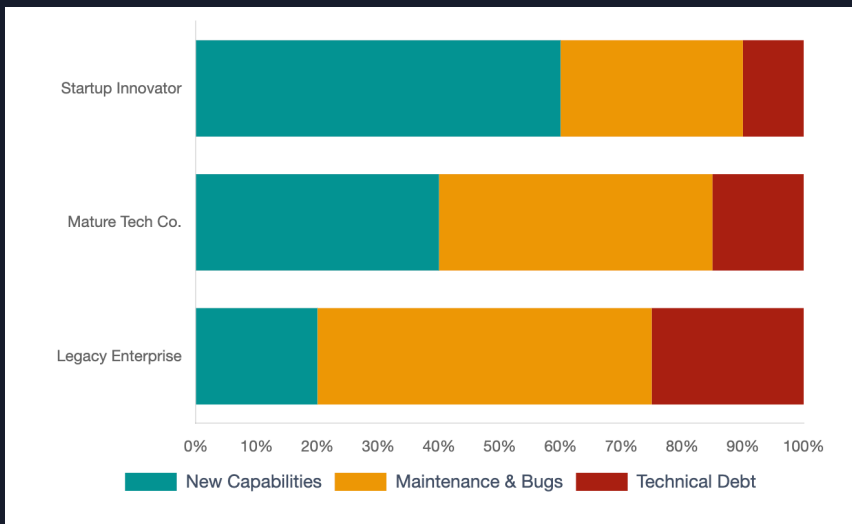
Deep Dive: The SPACE Framework

The 5 Dimensions of Productivity

- S Satisfaction & Well-being:** How happy and healthy developers are.
- P Performance:** The outcome and quality of the work produced.
- A Activity:** The count of development outputs (used with caution).
- C Communication & Collaboration:** How well teams work together.
- E Efficiency & Flow:** The ability to complete work without friction or delays.



Deep Dive: GetDX Core 4k



The Four Pillars of Impact



Speed: Measures development throughput and velocity.



Effectiveness: Assesses developer experience and work outcomes.



Quality: Measures software stability and reliability.



Impact: Directly connects engineering to business value and ROI.

Comparing the Developer Productivity Frameworks

DORA

Scope:

Narrow. The software delivery pipeline (commit to deploy).

Philosophy:

Prescriptive. A clear recipe of 4 key metrics.

Audience:

Technical Leaders. DevOps, SREs, Engineering Managers.

Data:

Quantitative. System data from CI/CD, Git, etc.

SPACE

Scope:

Broad. The entire socio-technical system.

Philosophy:

Flexible. A menu of dimensions to choose from.

Audience:

Engineering Management. Fosters empathetic conversations.

Data:

Hybrid. System data plus qualitative surveys.

GetDX Core 4

Scope:

Hybrid. Bridges engineering activity to business impact.

Philosophy:

Prescriptive. A unified recipe of 4 pillars.

Audience:

The Entire Org. A shared language for engineers and C-suite.

Data:

Hybrid. System data, surveys, and financial data.

The Three Pillars of World-Class DevEx

Fast, High-Quality Feedback Loops

Slow, ambiguous feedback is a primary source of frustration. The speed of the inner loop (local build/test) and outer loop (CI/CD, code review) is critical for maintaining momentum and iterating with confidence.

Low Cognitive Load

Human working memory is limited. When developers must wrestle with complex systems or poor documentation, less mental energy is available for creating solutions. High rework is a strong signal of high cognitive load.

Enabled "Flow State"

Flow, or being "in the zone," is where deep, creative work happens. It requires clear goals, immediate feedback, and protection from interruptions. It can take over 15 minutes to regain focus after a single interruption.

Fast, High-Quality Feedback Loops

1. Automated Visual Regression Testing in CI/CD
2. "Shifting Left" with Static Code Analysis and Linting in the IDE & Pre-Commit Hooks
3. Production "Canary" Deployments & Automated Monitoring with Meaningful Metrics
4. Daily "Mob Programming" or Pair Programming Sessions for Critical/Complex Tasks
5. Dedicated "Bug Bash" Weeks or Sprints with Stakeholder Involvement

Low Cognitive Load

1. Standardize Code Style & Linting Rules
2. Implement Version Control with Meaningful Commit Messages
3. Prioritize and Refactor Technical Debt Incrementally
4. Centralize Documentation and Knowledge Sharing
5. Implement Automated Testing at All Levels

Enabled "Flow State"

1. Implement a High-Signal Notification System & Prioritization
2. Adopt a Streamlined Code Review Process with Contextual Tooling
3. Standardize Development Environments and Automate Setup
4. Implement Short, Focused "Pomodoro" or Timeboxing Sessions
5. Cultivate a Culture of Psychological Safety and Open Communication

Avoid the Gamification Trap

Use Metrics for Improvement, Not Judgment

"when a measure becomes a target, it ceases to be a good measure."

- Goodhart's Law

So what do we measure?

- Cycle Time
- PR Review Time
- Rework Rate
- Meeting Load
- Time to First Commit
- Perceived Focus Time

Cycle Time

- Implement and enforce "Small Batch" Size Approach
- Optimize Code Review Process with Developer-Focused Tooling and Practices
- Invest in Test Automation and CI/CD Pipelines
- Improve Development Environment Setup and Standardization
- Proactively Identify and Remove Blocking Issues and Dependencies

PR Review Time

- Enforce "Small PR" Guidelines and Automation
- Implement a Reviewer Rotation and/or "Reviewer Roulette" System
- Mandate Clear and Concise PR Descriptions and Context
- Establish Service Level Agreements (SLAs) for PR Reviews and Make Them Visible
- Invest in Automated Code Analysis and CI/CD Integration

Rework Rate 🖋️

- Implement Comprehensive Code Reviews with a Focused Checklist
- Refine User Stories with Clearer Acceptance Criteria and Examples
- Invest in Better Tooling and Automation for Testing
- Implement a Robust Definition of Done (DoD) and Enforce It
- Improve Feedback Loops and Communication

Meeting Load

- Implement a "Meeting-Free Day" (or Half-Day) Policy
- Audit Meeting Invitations and Participation
- Standardize Meeting Agendas and Timeboxing
- Promote Asynchronous Communication Tools & Practices
- Implement a "Meeting Budget" or "Meeting Credit" System

Time to First Commit 🚀

- Provide Ready-to-Run Starter Projects/Templates
- Automate Environment Setup and Onboarding
- Simplify Code Contribution Process with Clear Guidelines and Tooling
- Offer Short, Focused "First Contribution" Tasks (aka "Good First Issues")
- Provide Active Mentorship and Support (paired with tooling)

Perceived Focus Time 💡

- Optimize Build Times with Incremental Builds and Caching
- Implement a "Quiet Period" During Deployments/Integrations
- Prioritize and Reduce Notification Overload
- Automate Repetitive Tasks with Scripting or Tools
- Improve Error Messaging and Debugging Tools

Build a Healthy Measurement Culture

- No framework is a silver bullet
- Continuous improvement, not judgment
- Communicate the 'why'
- Involve your team
- Focus on trends, not absolutes
- Combine quantitative data with qualitative human insights

Conclusion



Jeremy Meiss, Esq.

@jerdog.dev

If your company does not already have a process for gathering feedback (internal & external) on your product and/or the tools you use, you will not have a good Developer Experience ([#DevEx](#)), and I seriously question the commitment to it.

November 18, 2024 at 4:38 PM  Everybody can reply [↗](#)

DevEx is...

"ruthlessly eliminating barriers (and blockers) that keep your practitioners from being successful"

Thank you!



@jerdog.dev



/in/jeremyeiss



@jerdog



@jerdog@hachyderm.io



jmeiss.me



@AmJerdog

